15-462 Computer Graphics I
Lecture 4

Transformations

Vector Spaces
Affine and Euclidean Spaces
Frames
Homogeneous Coordinates
Transformation Matrices
[Angel, Ch. 4]

January 24, 2002
Frank Pfenning
Carnegie Mellon University

http://www.cs.cmu.edu/~fp/courses/graphics/

Announcement

- Guest lecture Tuesday, January 29
- *From Design to Production: How a Graphics Chip is Built*, Scott Whitman, nVidia

Compiling Under Windows (Answer)

- Must install GLUT
- Good source: http://www.opengl.org/
- Includes should be

```
#include <GL/glut.h>
#include <stdlib.h>
```

- Do not include <GL/gl.h> or <GL/glu.h>
- Run on lab machines before handing in!

Geometric Objects and Operations

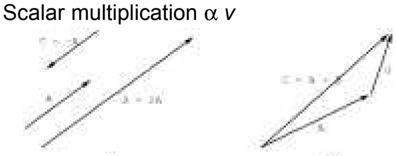
- Primitive types: scalars, vectors, points
- Primitive operations: dot product, cross product
- Representations: coordinate systems, frames
- Implementations: matrices, homogeneous coor.
- Transformations: rotation, scaling, translation
- Composition of transformations
- OpenGL transformation matrices

Scalars

- Scalars α, β, γ from a *scalar field*
- Operations $\alpha+\beta$, $\alpha \cdot \beta$, 0, 1, $-\alpha$, $(\ )^{-1}$
- “Expected” laws apply
- Examples: rationals or reals with addition and multiplication

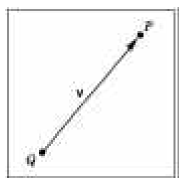
Vectors

- Vectors *u, v, w* from *vector space*
- Includes scalar field
- Vector addition *u + v*
- Zero vector **0**
- Scalar multiplication α *v*



Points

- Points *P, Q, R* from *affine space*
- Includes vector space
- Point-point subtraction $v = P - Q$
- Define also $P = v + Q$



Euclidean Space

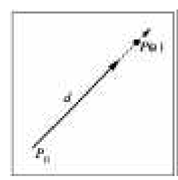
- Assume vector space over real number
- Dot product: $\alpha = u \cdot v$
- $\mathbf{0} \cdot \mathbf{0} = 0$
- *u, v* are *orthogonal* if $u \cdot v = 0$
- $|v|^2 = v \cdot v$ defines $|v|$, the *length* of *v*
- Generally work in an affine Euclidean space

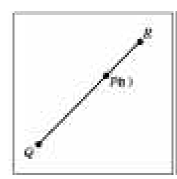
Geometric Interpretations

- Lines and line segments
- Convexity
- Dot product and projections
- Cross product and normal vectors
- Planes

Lines and Line Segments

- Parametric form of line: $P(\alpha) = P_0 + \alpha\, d$



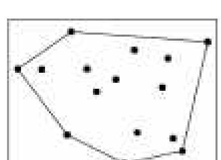


- Line segment between *Q* and *R*:
  $P(\alpha) = (1-\alpha)\, Q + \alpha\, R$ for $0 \leq \alpha \leq 1$

Convex Hull

- Convex hull defined by

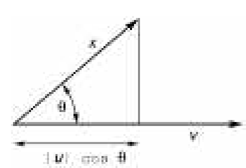
  $P = \alpha_1 P_1 + \cdots + \alpha_n P_n$
  for $a_1 + \cdots + a_n = 1$
  and $0 \leq a_i \leq 1$, $i = 1, ..., n$



Projection

- Dot product projects one vector onto other

  $u \cdot v = |u|\,|v| \cos(\theta)$



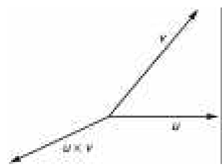
[diagram correction: x = u]

Normal Vector

- Cross product defines normal vector

  $u \times v = n$
  $|u \times v| = |u|\ |v|\ |\sin(\theta)|$

- Right-hand rule



Plane

- Plane defined by point P_0 and vectors u and v
- u and v cannot be parallel
- Parametric form: $T(\alpha, \beta) = P_0 + \alpha\, u + \beta\, v$
- Let $n = u \times v$ be the normal
- Then $n \cdot (P - P_0) = 0$ iff P lies in plane

Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Coordinate Systems

- Let v_1, v_2, v_3 be three linearly independent vectors in a 3-dimensional vector space
- Can write any vector w as

  $$w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$$

  for scalars $\alpha_1, \alpha_2, \alpha_3$
- In matrix notation:

$$\mathrm{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

Frames

- Frame = coordinate system + origin P_0
- Any point $P = P_0 + \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$
- Useful in with homogenous coordinates

Changes of Coordinate System

- Bases $\{u_1, u_2, u_3\}$ and $\{v_1, v_2, v_3\}$
- Express basis vectors u_i in terms of v_j

$$\begin{aligned} u_1 &= \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3 \\ u_2 &= \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3 \\ u_3 &= \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3 \end{aligned}$$

- Represent in matrix form

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \mathrm{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad \text{for} \quad \mathrm{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

Map to Representations

- $w = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3$, $\mathbf{a}^T = [\alpha_1\ \alpha_2\ \alpha_3]$
- $w = \beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3$, $b^T = [\beta_1\ \beta_2\ \beta_3]$

$$\mathrm{a}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = w = \mathrm{b}^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \mathrm{b}^T \mathrm{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

- So $\mathbf{a} = \mathbf{M}^T \mathbf{b}$ and $\mathbf{b} = (\mathbf{M}^T)^{-1} \mathbf{a}$
- Suffices for rotation and scaling, not translation

Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Linear Transformations

- 3×3 matrices represent linear transformations
  a = M b
- Can represent rotation, scaling, and reflection
- Cannot represent translation
- **a** and **b** represent vectors, not points

$$w = \mathrm{a}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Homogeneous Coordinates

- In affine space, $P = \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + P_0$
- Define $0 \cdot P = 0$, $1 \cdot P = P$
- Then

$$P = [\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad 1] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

- Point $\mathbf{p} = [\alpha_1\ \ \alpha_2\ \ \alpha_3\ \ 1]^T$
- Vector $w = \delta_1 v_1 + \delta_2 v_2 + \delta_3 v_3$
- Homogeneous coords: $\mathbf{a} = [\delta_1\ \ \delta_2\ \ \delta_3\ \ 0]^T$

Translation of Frame

- Express frame (u_1, u_2, u_3, P_0) in (v_1, v_2, v_3, Q_0)

$$\begin{aligned} u_1 &= \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3 \\ u_2 &= \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3 \\ u_3 &= \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3 \\ Q_0 &= \gamma_{41} v_1 + \gamma_{42} v_2 + \gamma_{43} v_3 + P_0 \end{aligned}$$

- Then

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \mathrm{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} \text{ for } \mathrm{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

Homogeneous Coordinates Summary

- Points $[\alpha_1\ \ \alpha_2\ \ \alpha_3\ \ 1]^T$
- Vectors $[\delta_1\ \ \delta_2\ \ \delta_3\ \ 0]^T$
- Change of frame

$$\mathrm{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Affine Transformations

- Translation
- Rotation
- Scaling
- Any composition of the above
- Express in homogeneous coordinates
- Need 4×4 matrices
- Later: projective transformations
- Also expressible as 4×4 matrices!

Translation

- **p**' = **p** + **d** where **d** = $[\alpha_x \ \ \alpha_y \ \ \alpha_z \ \ 0]^T$
- **p** = $[x \ \ y \ \ z \ \ 1]^T$
- **p**' = $[x' \ \ y' \ \ z' \ \ 1]^T$
- Express in matrix form **p**' = **T p** and solve for **T**

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling

- $x' = \beta_x x$
- $y' = \beta_y y$
- $z' = \beta_z z$
- Express as **p**' = **S p** and solve for **S**

$$\mathbf{S} = \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation in 2 Dimensions

- Rotation by θ about the origin
- $x' = x \cos \theta - y \sin \theta$
- $y' = x \sin \theta + y \cos \theta$
- Express in matrix form

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Note determinant is 1

Rotation in 3 Dimensions

- Decompose into rotations about x, y, z axes

$$\mathbf{R}_z = \begin{bmatrix} \cos\ \theta & -\sin\ \theta & 0 & 0 \\ \sin\ \theta & \cos\ \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\ \theta & -\sin\ \theta & 0 \\ 0 & \sin\ \theta & \cos\ \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

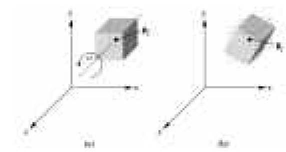
Compose by Matrix Multiplication

- $\mathbf{R} = \mathbf{R}_z \mathbf{R}_y \mathbf{R}_x$
- Applied from right to left
- $\mathbf{R}\,\mathbf{p} = (\mathbf{R}_z \mathbf{R}_y \mathbf{R}_x)\,\mathbf{p} = \mathbf{R}_z (\mathbf{R}_y (\mathbf{R}_x\, p))$
- "Postmultiplication" in OpenGL

Rotation About a Fixed Point

- First, translate to the origin
- Second, rotate about the origin
- Third, translate back
- To rotate by θ in about z around $\mathbf{p}_f$

  $\mathbf{M} = \mathbf{T}(\mathbf{p}_f)\, \mathbf{R}_z(\theta)\, \mathbf{T}(-\mathbf{p}_f) = \ldots$



Deriving Transformation Matrices

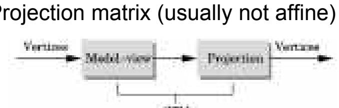
- Other examples: see [Angel, Ch. 4.8]
- See also Assignment 2 when it is out
- Hint: manipulate matrices, but remember geometric intuition

Outline

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

Current Transformation Matrix

- Model-view matrix (usually affine)
- Projection matrix (usually not affine)



- Manipulated separately

```
glMatrixMode (GL_MODELVIEW);
glMatrixMode (GL_PROJECTION);
```


Manipulating the Current Matrix

- Load or postmultiply

```
glLoadIdentity();
glLoadMatrixf(*m);
glMultMatrixf(*m);
```

- Library functions to compute matrices

```
glTranslatef(dx, dy, dz);
glRotatef(angle, vx, vy, vz);
glScalef(sx, sy, sz);
```

- Recall: last transformation is applied first!

Summary

- Vector Spaces
- Affine and Euclidean Spaces
- Frames
- Homogeneous Coordinates
- Transformation Matrices
- OpenGL Transformation Matrices

OpenGL Tutors by Nate Robins

- Run under Windows
- Available at http://www.xmission.com/~nate/tutors.html
- Example: Transformation tutor